PB-239 896

ROUTING OF SOLID WASTE COLLECTION VEHICLES

APPENDIX A: A LINEAR PROGRAMMING APPROACH FOR THE TRAVELING SALESMAN PROBLEM

ILLINOIS UNIVERSITY

PREPARED FOR
NATIONAL ENVIRONMENTAL RESEARCH CENTER

DECEMBER 1974

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

TECHNICAL REPORT DATA
(Please read Instructions on the reverse before completing)

1. REPORT NO. EPA-670/2-74-035a	2.	3. PB 239 896
4. TITLE AND SUBTITLE ROUTING OF SOLID WASTE COLLECTION VEHICLES; First Annual Report and Appendix A: A Linear Programming Approach for the Traveling Salesman Problem		5. REPORT DATE December 1974-Issuing Date 6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) Jon C. Liebman and Saman Hong		8. PERFORMING ORGANIZATION REPORT NO.
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Illinois Urbana, Illinois 61801		10. PROGRAM ELEMENT NO. 1DB314;ROAP 02AAE;TASK 04 11. CONTRACT/GRANT NO. R801289
12. SPONSORING AGENCY NAME AND ADDRESS National Environmental Research Center Office of Research and Development U.S. Environmental Protection Agency Cincinnati, Ohio 45268		13. TYPE OF REPORT AND PERIOD COVERED First Annual Report 14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

See EPA-670/2-74-035b, EPA-670/2-74-036a, and EPA-670/2-74-036b

16. ABSTRACT

The first report provides the background for routing solid waste collection vehicles. Problems encountered with existing methods and possible solutions are given. Appendixes A and B are solutions to two classic routing problems. Appendix A takes the integer programming formulation of Dantzig, Fulkerson, and Johnson which is relaxed to a linear program and a column generation scheme found for its dual. A matching constraint is then added to eliminate some of the fractional solutions which contain an odd cycle of noninteger edges. Finally, a branch-and-bound scheme is employed to search for an optimal all-integer solution. Appendix B (separately bound) is a study applying network theory to the problem of routing a solid waste collection vehicle through a street network. The total length of the tour is sought rather than the detailed and chronologic schedule of the vehicle. This tour is found by minimizing the total length of street segments which have to be retraced. The problem can be described mathematically as a linear program whose optimal solution is guaranteed to be integer. Due to the size of the problem, linear program algorithms cannot be used.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
*Collection, *Routing, *Computer programming, Management, Computers, Refuse, *Vehicles, Mathematical models, Linear programming	Solid waste, Solid waste management, *Solid waste collection, Mathematical solutions, *Computer routing	13B

18. DISTRIBUTION STATEMENT Release to public	19. SECURITY CLASS *(This Report)* UNCLASSIFIED	21. NO. OF PAGES 66
	20. SECURITY CLASS *(This page)* UNCLASSIFIED	22. PRICE 4.25

EPA Form 2220-1 (9-73)

Reproduced by
NATIONAL TECHNICAL INFORMATION SERVICE
U.S. Department of Commerce
Springfield, VA. 22151

PRICES SUBJECT TO CHANGE

071057

EPA-670/2-74-035a
December 1974

ROUTING OF SOLID WASTE COLLECTION VEHICLES

First Annual Report

and

Appendix A: A Linear Programming Approach

for the Traveling Salesman Problem

By

Jon C. Liebman, Principal Investigator,
and Saman Hong
University of Illinois
Urbana, Illinois 61801

Program Element No. 1DB314

Project Officer

Donald A. Oberacker
Solid and Hazardous Waste Research Laboratory
National Environmental Research Center
Cincinnati, Ohio 45268

NATIONAL ENVIRONMENTAL RESEARCH CENTER
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
CINCINNATI, OHIO 45268

ia

REVIEW NOTICE

The National Environmental Research Center--Cincinnati has reviewed this report and approved its publication. Approval does not signify that the contents necessarily reflect the views and policies of the U.S. Environmental Protection Agency, nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

FOREWORD

Man and his environment must be protected from the adverse effects of pesticides, radiation, noise and other forms of pollution, and the unwise management of solid waste. Efforts to protect the environment require a focus that recognizes the interplay between the components of our physical environment--air, water, and land. The National Environmental Research Centers provide this multidisciplinary focus through programs engaged in

- studies on the effects of environmental contaminants on man and the biosphere, and
- a search for ways to prevent contamination and to recycle valuable resources.

As part of these activities, this report provides the background for routing solid waste collection vehicles, the problems encountered with existing methods, and possible solutions.

A. W. Breidenbach, Ph.D.
Director
National Environmental
Research Center, Cincinnati

TABLE OF CONTENTS

I.	Routing Problems	1
II.	Routing Methods	2
III.	Research Progress	5
	A. Traveling Salesman Approaches	6
	B. The Chinese Postman's Problem on a Mixed Network	6
IV.	Continuing Research	7
	Appendix A: A Linear Programming Approach for the Traveling Salesman Problem	8
	Appendix B:	Separately Bound

First Annual Report

Jon C. Liebman, Principal Investigator

I. Routing Problems

Collection vehicle routing problems occur in two different ways. On infrequent occasions, municipalities open new solid waste disposal or treatment facilities, requiring the re-routing of at least a large portion of the entire collection fleet. More frequently, the continuing processes of population shift, new construction, demolition, urban renewal and other land use changes will result in the need to add new routes, delete old routes or re-route small portions of the collection fleet.

In most cities, a rather large effort is devoted to the routing required when new facilities are opened. This effort requires a significant amount of time and usually leads to solutions which are fairly economical. Commonly, however, frequent needs for minor re-routing are ignored until the situation becomes serious. Thus, the "good" routes determined initially gradually deteriorate in efficiency as the city changes, until a large study is found to be necessary to restore the earlier economies.

The overall routing problem may be viewed as consisting of two parts. The first, hereafter called the districting problem, is to divide the area to be serviced into individual districts or sectors, each of which contains exactly one truckload of waste. The second part of the problem is to route each collection vehicle optimally from depot to district, through the district, and from district back to depot. Obviously, the two problems are not independent, as the proper selection

of district boundaries has a large effect on the lengths of the routes within the districts and between the districts and the depot.

A "good" solution to a districting-routing problem is one which results in a low cost of operation of the system. This cost is a function of the time required to cover all of the routes and the distance traveled over the routes. For simplicity in the early stages of this research, the objective considered has been the minimization of travel distance; but in all cases it is understood that travel distance could be replaced by a weighted sum of distance and time.

The aim of this research is to develop methods which will aid in the solution of routing and districting problems. These methods may include large computer programs which solve both districting and routing problems optimally (or near-optimally), as well as smaller computer programs which interact with decision-makers, and even simple rules which may be followed to design routes manually.

II. Routing Methods

Problems of vehicle routing have been considered in the operations research literature for some time. Perhaps the oldest problem in the literature, and one which has had a great deal of research in solution techniques, is the traveling salesman problem. In this problem, a group of service points must be visited by a single vehicle, which must start and finish its tour at the same point (depot). The route which visits each service point with a minimum total distance is to be determined. There are many solution methods for this problem, but none can be considered computationally feasible for problems with more than about

100 service points. It should be noted that the solid waste collection vehicle routing problem (after districts have been established) can be cast in the form of a traveling salesman problem by treating each pickup point as a service point. For most daily routes, the number of service points in such a problem will be significantly larger than 100, so that it is not feasible to solve the collection vehicle routing problem in this manner.

There are a number of variations on the traveling salesman problem which include some form of districting and the use of more than one vehicle. These are variously called the M-salesman problem, the cloverleaf problem, the truck dispatching problem and other names in the literature. In almost all cases, however, they depend for optimal solution on a method for solving the traveling salesman problem, and thus are not of particular use in this research. However, there are some heuristic or approximate solution methods for some of these problems which might prove useful.

It is also possible to use the traveling salesman approach by aggregating pickup points. Thus, a non-optimal and approximate solution might be found by using entire block-faces as single service points in a traveling salesman formulation. A great deal of aggregation would be necessary, however, in order to reduce the problem size sufficiently to make city-wide solutions feasible.

A second and much more recent general approach to vehicle routing is the Chinese postman's problem*. Here it is assumed that the demands for service are uniformly distributed along the streets in the network, rather than only at specific points. In order to provide the required

*So named because the first formulation of the problem referred to a postman's route and appeared in the Journal of Chinese Mathematics.

service, it is necessary to traverse every street at least once. As before, the single service vehicle is required to start and finish at the same depot, and a minimum distance tour is desired. Clearly, the minimum distance tour has a length at least as great as the sum of the lengths of all of the streets, since each must be traversed at least once. Euler in 1736 proved that a tour which traverses each street exactly once is possible only when every intersection is touched by an even number of streets. When this is not the case, some streets must be retraced; these retracings are the equivalent in the solid waste collection vehicle problem to traveling over a street without collecting, and it is this retracing distance which must be minimized.

A particularly elegant mathematical algorithm for solving the Chinese postman's problem* was found by Edmonds in 1965. His method is readily adaptable to digital computers and will solve extremely large problems in a very short time. There is, however, one additional requirement which the street network must meet in order to be able to use Edmonds' method. No one-way streets are permitted in the network. This requirement makes it impossible to use the Edmonds method directly in many urban residential areas.

Edmonds has also pointed out that if all of the streets in the network are one-way streets, then there is a particularly simple computer algorithm for finding the postman solution. In cities where collection is done only on the right side of the street (thus requiring the collection vehicle to traverse both sides of each street), each two-way

*The method actually finds those streets which must be retraced; a separate (and simple) algorithm must then be used to generate a tour which retraces the proper streets.

street may be regarded as two one-way streets, and this approach will yield optimal solutions.

Unfortunately, in most cities there is a mixture of one-way and two-way streets and there has been no known method for solving the minimum routing problem in such a mixed network. A new, as yet unpublished paper by Edmonds and Johnson suggests a heuristic solution of the problem which has never been tested.

An optimal solution to the Chinese postman's problem would provide optimal routes for collection vehicles if the districts are already known. Such a solution is useful for the small scale minor re-routing problem. In addition, this solution could be coupled with either an optimal or a heuristic districting algorithm to produce a solution for the larger problem.

III. Research Progress

At the beginning of the research, several areas for investigation during the first year of the project were specified. These are:

a) Further investigation of traveling salesman approaches

b) Solution of the Chinese postman's problem on a mixed network

c) Initial investigation of heuristic approaches for districting and for solving the multiple postman problem.

Of these areas, it was decided to concentrate most heavily on solving the Chinese postman's problem on a mixed network, since this solution seems to be crucial to any general routing solution.

Work on projects a) and b) has been completed and is described in the following paragraphs. Work on project c) is continuing and will be discussed briefly in the last section of this report.

A. Traveling Salesman Approaches

Although it was felt from the beginning that the traveling salesman form of the routing problem would not be the most promising approach, partial support was provided by this grant for research into a new algorithm for solution of the traveling salesman problem and the M-salesman problem. The research has culminated successfully in that the new algorithm has proved to be computationally efficient for solving some traveling salesman problems. However, although the results are of theoretical significance in the study of combinatorial problems, the method is not sufficiently efficient to permit practical solution of reasonable sized M-salesman problems.

A complete and detailed description of the research on this algorithm is contained in Appendix A of this report.

B. The Chinese Postman's Problem on a Mixed Network

The major thrust of the research during the first year has been on the solution of this problem. It was intended first to investigate optimal algorithms and then to explore heuristic algorithms if no good optimal algorithm was developed. Fortunately, a complete mathematical method which determines optimal postman's routes through a network of both directed and undirected streets has been developed. On an IBM 7094 computer the algorithm has solved networks of 46 intersections and 75 streets in between 0.24 minutes and 3.13 minutes. It appears that the running time is greatest when a large number of the streets are undirected (two-way). The solution times are practical for common collection vehicle routing problems. In addition, the same algorithm

will also provide solutions when all streets are two-way, or when all streets are one-way. For these special cases, however, the algorithm is inefficient when compared to the existing special methods for the solution.

The development of this algorithm permits the solution of all single vehicle routing problems to obtain minimum distance routes.* This represents a major step forward in the construction of useful techniques for general collection vehicle routing. If a suitable districting method can be devised and coupled with this algorithm, the theoretical framework for complete solutions will be complete.

The complete algorithm, together with a proof of its optimality and a discussion of computational experience, is contained in Appendix B.

IV. Continuing Research

Solution of the multiple postman problem is the utlimate aim of this research, since it provides a complete method for routing collection vehicles. The emphasis of the work is on the development of heuristic districting methods which can be coupled with optimal single-vehicle routing algorithms. The research in the second year will be centered on this goal, together with peripheral projects aimed at coding the existing algorithms for ease of use.

*As pointed out earlier, "distance" may be replaced with some combination of distance and time.

Appendix A: A Linear Programming Approach for the Traveling Salesman Problem

By Saman Hong

ABSTRACT

The integer programming formulation of Dantzig, Fulkerson and Johnson is relaxed to a linear program and a column generation scheme is found for its dual. A matching constraint is then added to eliminate some of the fractional solutions if the linear program terminates in noninteger solutions which contain an odd cycle of noninteger edges. Finally, a branch-and-bound scheme is employed to search for an optimal all-integer solution.

Two types of multi-salesmen problems are also introduced and transformed into single-salesman problems.

PREFACE

The study of techniques for routing solid waste collection vehicles, of which this paper constitutes a portion of the first annual report, is supported by grant EP 801289 from the Solid Waste Research Division of the U.S. Environmental Protection Agency.

The body of this paper is the doctoral dissertation of Dr. Saman Hong, with minor editorial revision. The dissertation was submitted to The Johns Hopkins University. Dr. Hong is currently at Bernard Baruch College, City College of New York.

The assistance of Dr. Mandell Bellmore and Dr. Jack Elzinga of The Johns Hopkins University is acknowledged with thanks.

Jon C. Liebman
Principal Investigator

TABLE OF CONTENTS

Chapter 1	Introduction	11
Chapter 2	The Traveling Salesman Problem	13
2.1	Preliminary Notions of Linear Graphs	13
2.2	The Traveling Salesman Problem on a Graph	14
2.3	m-Salesman Problem	17
Chapter 3	The Integer Programming Formulation and its Relaxed Linear Program	26
3.1	0-1 Programming Formulation	26
3.2	The Relaxed Linear Program	29
3.3	The Dual Linear Program	31
3.4	A Column Generation Scheme	36
Chapter 4	A Search for an All-Integer Solution	39
4.1	A Matching Constraint for an Odd Cycle	39
4.2	A Branch-And-Bound Scheme	45
Chapter 5	Computational Results and Conclusions	57
References		59

CHAPTER 1

INTRODUCTION

In the traveling salesman problem a salesman is to visit each of a set of given points (called nodes) exactly once and to return to the point of his origin. His route is called a tour and the objective is to find a tour with a minimum total distance. The distance between any pair of points is known.

The problem is called symmetric if the distance from point i to point j is equal to the distance from point j to point i for every pair of points i and j; otherwise, it is called asymmetric.

The traveling salesman problem has long been considered one of the fundamental combinatorial problems for which a solution method would facilitate that of many others. The family of algorithms for the traveling salesman problem is large, but none of them works well for large scale problems.

A survey of algorithms developed through 1966 is given by Bellmore and Nemhauser[2]. Two more algorithms for the traveling salesman problem have been developed recently.

In 1969 Bellmore and Malone[1] designed an algorithm by imbedding the 2-matching algorithm of Edmonds[5] in a branch-and-bound scheme. They reported that the expected computer time would grow as a polynomial of degree 6 or less in the number of nodes.

Held and Karp[10][11] completed an improved branch-and-bound algorithm for the symmetric traveling salesman problem in 1970, in which they used an "ascent method" to obtain a lower bound. The ascent method is an iterative method of computing or approximating the greatest lower bound.

The reported computational results show that their algorithm is the most efficient one so far for the symmetric problem.

An integer programming formulation of the traveling salesman problem was demonstrated as early as 1954 by Dantzig, Fulkerson and Johnson[3]. However, the number of constraints in their formulation was so enormous (on the order of 2^n, where n is the number of nodes) that even the relaxed linear program could not be solved in general. The main purpose of this research has been to develop an algorithm to solve the integer programming formulation of Dantzig, et al.

In Chapter 2, the fundamental concepts of graph theory are introduced and the traveling salesman problem is redefined in terms of a graph. Two types of multi-salesmen problems are also introduced and transformed into single-salesman problems.

In Chapter 3, the integer programming formulation of Dantzig et al. is presented. It is then relaxed to a linear program and the dual of the linear program is taken. Finally, a column generation scheme for the dual is discussed.

In Chapter 4, a matching constraint is introduced to eliminate some fractional solutions of the relaxed linear program, and a branch-and-bound scheme is presented in order to search for an optimal all-integer solution.

Computational results are presented in Chapter 5.

CHAPTER 2

THE TRAVELING SALESMAN PROBLEM

2.1 Preliminary Notions of Linear Graphs

A directed linear graph G(N,A) consists of a collection N of elements 1, 2, 3, ..., n, together with a subset A of the ordered pairs (i,j) of the elements taken from N. It is assumed throughout that N is a finite set. The elements of N are called nodes and the elements of A are called arcs. Node i is called the transmitter of arc (i,j) and node j is called the receiver of arc (i,j). Arc (i,j) is said to be directed from node i to node j.

Suppose there is a real number d_{ij} associated with each arc (i,j) in a graph G(N,A). We call d_{ij} the distance of arc (i,j) and denote the graph with d_{ij} by G(N,A;D). A graph G(N,A;D) is called symmetric if $d_{ij} = d_{ji}$ for every arc (i,j) and is called asymmetric otherwise. Figure (2.1) is an example of an asymmetric graph having six nodes and ten arcs. The number of each arc represents the distance of the arc.

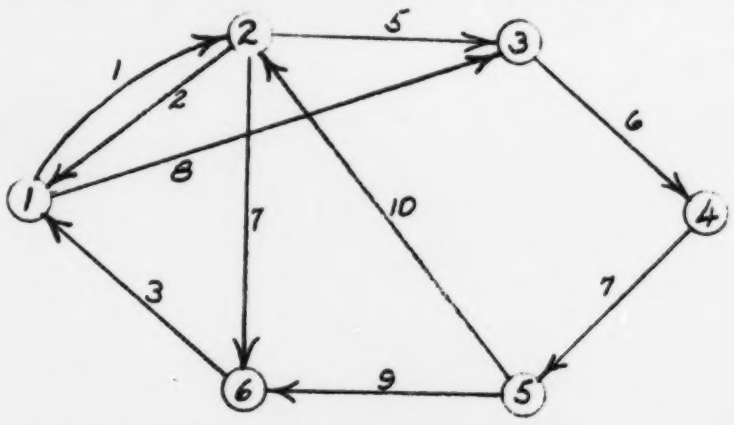


Fig. (2.1) An example of an asymmetric graph

A graph G(N,A;D) is called a Euclidean graph if each node of the graph is realizable as a point on a plane with d_{ij} being given by the Euclidean distance between point i and point j.

A sequence $i_1, i_2, \ldots, i_p$ of distinct nodes is called a chain from node i_1 to node i_p if there is an arc directed from each node to its successor in the sequence. If the first node and the last node of a chain are identical, that is, if $i_1 = i_p$, then the chain is called a cycle.

If a cycle contains an even (odd) number of nodes, then the cycle is called an even (odd) cycle. If any node is common to two cycles then these two cycles are said to be node-joint. If a cycle contains every node of the graph, then the cycle is called a tour; otherwise, it is called a subtour.

A tour is cyclic in the sense that any node can be the starting node, and we will fix node 1 as the starting node of any tour. This starting node will be called the home node, and the other nodes will be called the customer nodes.

If the direction of every arc is unspecified, then the graph is said to be undirected. An undirected graph will be denoted by G(N,E;D) to distinguish it from a directed graph G(N,A;D). The elements of E are the unordered pairs of nodes and are called edges. The concept of a tour or a cycle is the same in an undirected graph as in a directed graph except that the edges must replace the arcs.

2.2 The Traveling Salesman Problem on a Graph

The traveling salesman problem on a graph is to find a tour with a minimal total distance, that is, to

$$\text{minimize } z(t) = \Sigma_{(i,j)\in t} d_{ij} \qquad (2\text{-}1)$$

where t represents a tour and the summation is over the arcs directed from a node to its successor in the tour. Such arcs are simply said to be "in the tour t," and z(t) is called the distance of the tour t.

If one can find a minimal distance tour, then one can also find a minimal distance route along which each node is passed through at least once. This comes from a theorem by Hardgrave and Nemhauser[9]. Here we quote the theorem from Bellmore and Nemhauser[2].

> "Theorem 1
>
> If D satisfies the triangle inequality*, then there is an optimal route along which each node is passed through once and only once."

In other words, if the distances satisfy the triangle inequality, then no node need be passed through more than once to minimize the traveling distance along the route. On the other hand, if the triangle inequality is not satisfied, then some of the nodes may be passed through more than once along the minimal distance route.

Note that the triangle inequality is always satisifed if d_{ij} is replaced by d'_{ij}, where d'_{ij} is the distance of a shortest chain from node i to node j. Thus, the minimal distance route on G(N,A;D) can be found by obtaining a minimal distance tour on a transformed graph G(N,A;D'). If (i_p,i_q) is an arc in the optimal tour under D' and if $(i_p, i_s, i_t, \ldots, i_q)$ is a shortest chain from node i_p to node i_q under D, then the optimal route

---

* The triangle inequality is:

$$d_{ij} \leq d_{ik} + d_{kj}$$

under D contains the sequence $(i_p, i_s, i_t, \ldots, i_q)$. Therefore, we will restrict ourselves to the case where every node must be passed through exactly once.

The distance d_{ij} can be any real number, but without loss of generality, we can assume that all d_{ij}'s are nonnegative. This is because the traveling salesman problem on a graph with negative distances can always be transformed into an equivalent problem on the same graph with nonnegative distances as the following theorem shows.

Theorem 2.1

Given a graph G(N,A;D), let D be transformed into D' by:

$$d'_{ij} = d_{ij} + h_i + h_j \qquad (2\text{-}2a)$$

where h_i is a real number associated with node i. Then

$$z'(t) = z(t) + 2 \sum_{i \in N} h_i \quad \text{for every tour } t \qquad (2\text{-}2b)$$

where z(t) is the distance of the tour t under D and z'(t) is the distance of the same tour t under D'.

(Proof)

In any tour, there are exactly two arcs incident with each node. Therefore, h_i is added exactly twice, for every node i, to the distance of the tour under the transformation.

Q.E.D.

Therefore, if $d_{ij} < 0$ for some arc (i,j), then by taking very large numbers for h_i and h_j, d'_{ij} can be made nonnegative, and the problem remains unchanged.

There are many problems which can be transformed into the traveling salesman problem. One such example is the machine scheduling problem in

which nodes correspond to the jobs to be performed and the distance d_{ij} is the machine set-up time for job j after job i is completed. (See Gilmore and Gomory[6].)

2.3 m-Salesmen Problem

In this section we introduce a multi-salesmen problem in which there are m salesmen at the home node and each customer node must be visited by one of the m salesmen. Here, we consider two different cases:

(i) every one of the m salesmen must visit at least one customer node,

(ii) some of the m salesmen can be idle and there is a cost c_i of using the i^{th} salesman which must be added to the objective function if the i^{th} salesman is used.

Case (i) is equivalent to finding a set of m cycles joined at the home node with minimal total distance. Here, every customer node must belong to exactly one of the m cycles. We will call such a set of m cycles an m-tour.

A sequence of nodes representing an m-tour is obtained by arranging the m sequences of nodes each of which represents one of the m cycles. In the sequence of nodes representing an m-tour, every customer node appears exactly once, while the home node appears m+1 times, once as the starting node, once as the terminal node and m-1 times as an intermediate node.

In the following theorem 2.2, we will prove that the minimum distance m-tour on a graph $G(N,A;D)$ can be found by obtaining a minimum distance tour on a transformed graph $G(N^m,A^m;D^m)$. The set of nodes, N^m, in the transformed graph is obtained by adding m-1 artificial nodes, called pseudo-nodes, to N. The set of arcs, A^m, is obtained by adding to A the

following arcs of (a) and (b), called pseudo-arcs, with the specified distances for the additional entries in D^m.

(a) if $(1,j)$ is an arc in A, then add arc (p_i,j) to A with the distance $d_{p_ij} = d_{1j}$ for every pseudo-node p_i and for every customer node j,

(b) if $(j,1)$ is an arc in A, then add arc (j,p_i) to A with the distance $d_{jp_i} = d_{j1}$ for every pseudo-node p_i and for every customer node j,

where, node 1 is assumed to be the home node.

Theorem 2.2

Given an m-tour in $G(N,A;D)$, there is a tour in $G(N^m,A^m;D^m)$ with an equal distance, and vice versa.

(Proof)

Suppose an m-tour is given. The home node appears m+1 times in the sequence of nodes representing the m-tour (once as the starting node, once as the terminal node, and m-1 times as an intermediate node). Replace each of the m-1 intermediate home nodes by one of the pseudo-nodes. Then the resulting sequence represents a tour in $G(N^m,A^m;D^m)$. When the home node is replaced by a pseudo-node p_i, arc $(j,1)$ and arc $(1,k)$ are replaced by pseudo-arcs (j,p_i) and (p_i,k), where node j is the predecessor of the replaced home node and node k is the successor to the replaced home node in the sequence. The distance, however, remains unchanged because $d_{p_ij} = d_{1j}$ and $d_{jp_i} = d_{j1}$ for every pseudo-arc (p_i,j) or (j,p_i).

Given a tour in $G(N^m,A^m;D^m)$, an m-tour in $G(N,A;D)$ with the same distance can be obtained by reversing the above process, that is, by replacing the pseudo-nodes by the home node. Q.E.D.

Figure (2.2) shows an example of G(N,A;D) together with a 2-tour, transformed into $G(N^2,A^2;D^2)$ together with a corresponding tour.

Case (ii) can be solved by solving the r-salesman problem of case (i) m times for r = 1, 2, ..., m, as the following theorem shows.

Theorem 2.3

Let z^r be the distance of a minimal distance r-tour on $G(N^r,A^r;D^r)$ for r = 1, 2, ..., m. Let the costs of salesmen be arranged such that

$$c_1 \leq c_2 \leq \cdots \leq c_m \tag{2-3}$$

Then, the case (ii) is equivalent to solving

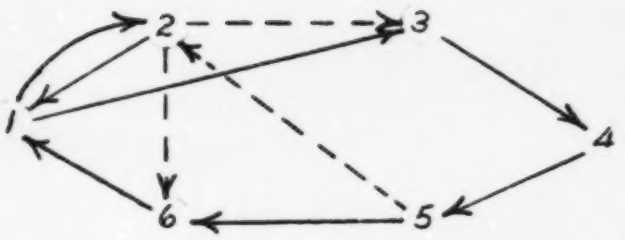
$$\underset{1 \leq r \leq m}{\text{minimize}} \{z^r + \Sigma_{i=1}^{r} c_i\} \tag{2-4}$$

(Proof)

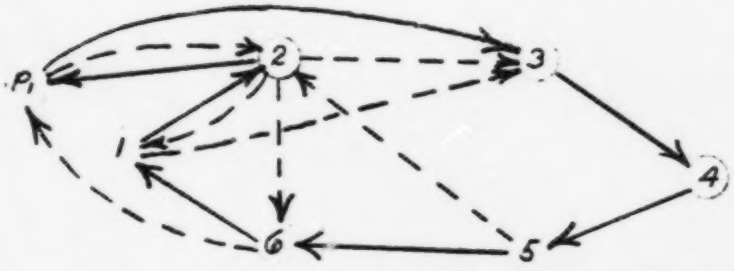
The minimal distance r-tour is independent of which r salesmen are used and $\Sigma_{i=1}^{r} c_i$ is the minimal cost of using r of the m salesmen by equation (2-3). Therefore, $z^r + \Sigma_{i=1}^{r} c_i$ is the best completion when r salesmen are used and (2-4) solves the m-salesmen problem of case (ii).

Q.E.D.

An alternative and better way to solve case (ii) is to find a minimal distance tour on a transformed graph $G(N^m,\underline{A}^m;\underline{D}^m)$ as will be proved in theorems 2.4 and 2.5. N^m is the same as before, that is, N plus m-1 pseudo-nodes p_1, p_2, ..., p_{m-1}, and $\underline{A}^m$ is obtained by adding to A the following arcs of (a) through (d) with specified distances for the additional entries in $\underline{D}^m$.



(a) A graph G(N,A;D) with a 2-tour



(b) A transformed graph $G(N^m,A^m;D^m)$ with the corresponding tour for m = 2.

Fig. (2.2)

(a) If (1,j) is an arc in A then add arc (p_i,j) to A with distance

$$d_{p_i j} = d_{1j} + (1/2)c_i \qquad (2\text{-}5a)$$

for every pseudo-node p_i and for every customer node j.

(b) If (j,1) is an arc in A, then add arc (j,p_i) to A with distance

$$d_{jp_i} = d_{j1} + (1/2)c_i \qquad (2\text{-}5b)$$

for every pseudo-node p_i and for every customer node j.

(c) Add arcs (p_i,p_j) and (p_j,p_i) to A with distances

$$d_{p_i p_j} = -\tfrac{1}{2}c_i + \tfrac{1}{2}c_j \qquad (2\text{-}6a)$$

$$d_{p_j p_i} = -\tfrac{1}{2}c_j + \tfrac{1}{2}c_i \qquad (2\text{-}6b)$$

for every pair of pseudo-nodes p_i and p_j.

(d) Add arcs $(1,p_i)$ and $(p_i,1)$ to A with distances

$$d_{1p_i} = -\tfrac{1}{2}c_m + \tfrac{1}{2}c_i \qquad (2\text{-}7a)$$

$$d_{p_i 1} = -\tfrac{1}{2}c_i + \tfrac{1}{2}c_m \qquad (2\text{-}7b)$$

for every pseudo-node p_i.

In addition, distances d_{1j} and d_{j1} must be changed to

$$d_{1j} \leftarrow d_{1j} + \tfrac{1}{2}c_m \qquad (2\text{-}8a)$$

$$d_{j1} \leftarrow d_{j1} + \tfrac{1}{2}c_m \qquad (2\text{-}8b)$$

for every arc $(1,j)$ and $(j,1)$, where, as before, node 1 is assumed to be the home node.

Figure (2.3) shows an example of $G(N^m,\underline{A}^m;\underline{D}^m)$ transformed from $G(N,A;D)$ of Figure (2.1) for $m = 2$.

Theorem 2.3

Given an r-tour t_r with distance $z(t_r)$ in $G(N,A;D)$ using salesmen i_1, i_2, ..., i_r, there exists a tour t in $G(N^m, \underline{A}^m;\underline{D}^m)$ for which the distance $z''(t)$ is equal to

$$z''(t) = z(t_r) + \Sigma_{k=i_1}^{i_r} c_k \qquad (2\text{-}9)$$

where c_k is the cost of using salesman k for $k = 1, 2, \ldots, m$.

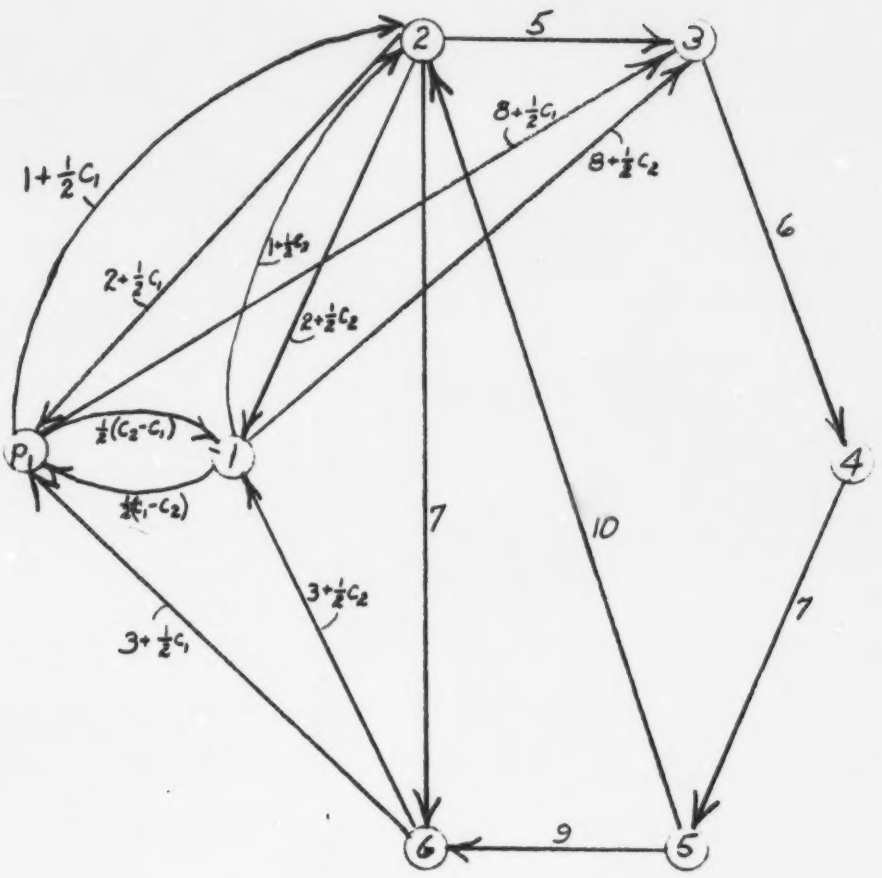


Fig. (2.3) A graph $G(N^m, \underline{A}^m; \underline{D}^m)$ transformed from G(N,A;D) of Fig. (2.1) for m = 2

(Proof)

An r-tour is made of r cycles,each of which is assigned to one of the m salesmen. The following steps generate a tour in $G(N^m,\underline{A}^m;\underline{D}^m)$ from an r-tour in G(N,A;D).

(Step 1) Make a sequence of nodes which represents the given r-tour. Here, if one of the r cycles is assigned to salesman m, then let the nodes in this cycle be the first part of the sequence. As an example, consider a 4-salesmen problem. If a 2-tour is given as in Figure (2.4), then the sequence for the 2-tour is given as:

1, 2, 1, 3, 4, 5, 6, 1

(Step 2) If the successor of any home node in the sequence belonged to the cycle assigned to salesman i for some i = 1, 2, ..., m-1, then replace that home node by a pseudo-node p_i. The sequence of the example would change to:

p_1, 2, p_3, 3, 4, 5, 6, 1

(Step 3) If the first node of the resulting sequence is not the home node, then locate the home node at the beginning of the sequence. This occurs if salesman m is not used for the r-tour. The sequence of the example would change to:

1, p_1, 2, p_3, 3, 4, 5, 6, 1

(Step 4) Place all the pseudo-nodes which are not in the sequence before the last node of the sequence. Then the resulting sequence is a tour in $G(N^m,\underline{A}^m;\underline{D}^m)$. The sequence of the example would change to:

1, p_1, 2, p_3, 3, 4, 5, 6, p_2, 1

and the corresponding tour is shown in Figure (2.5).

If salesman i is used for the r-tour, then c_i is added to the distance of the tour in step 2 by equations (2-5a) and (2-5b), for i = 1, 2, ..., m-1. If salesman m is used, then c_m is added to the distance of the tour in step 1 or step 3 by (2-8a) and (2-8b). When the unused pseudo-nodes are added to the sequence in step 4, the distance of the tour remains unchanged because of (2-6a), (2-6b), (2-7a), and (2-7b). Therefore, equation (2-9) is satisfied.

Q.E.D.

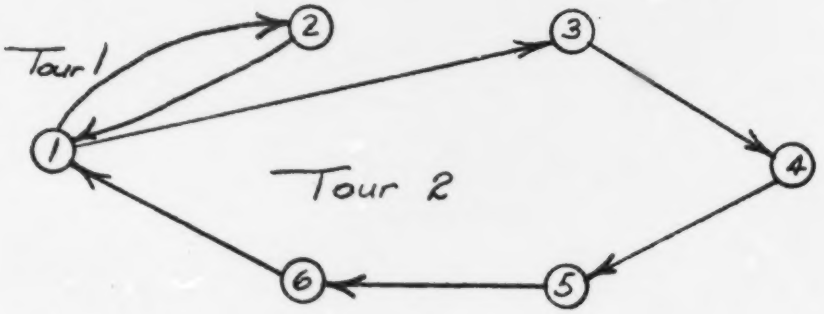


Fig. (2.4) A 2-tour for a 4-salesmen problem

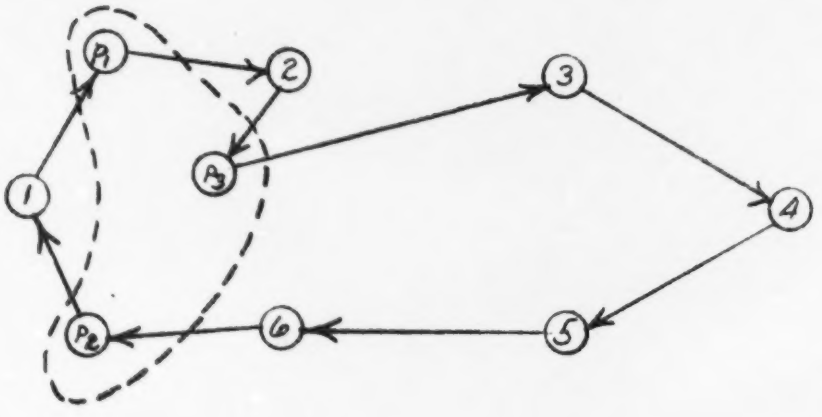


Fig. (2.5) A 4-salesmen tour obtained from a 2-tour of Fig. (2.4)

Theorem 2.5

Given a tour in $G(N^m, \underline{A}^m; \underline{D}^m)$ there is an r-tour in G(N,A;D) such that equation (2-9) is satisfied for some number r.

(Proof)

The following steps generate an r-tour from the given tour such that (2-9) is satisfied.

(Step 1) Make a sequence of nodes which represents the given tour.

(Step 2) If a pseudo-node or the home node is followed by the home node or a pseudo-node in the sequence, then discard such a pseudo- or home node from the sequence.

(Step 3) Replace the remaining pseudo-nodes in the sequence by the home node.

Then the resulting sequence is an r-tour in G(N,A;D). When the pseudo-node p_i is replaced by the home node in step 3, c_i is deducted from the distance of the tour and (2-9) is satisfied at the end. The number r is equal to the number of pseudo-nodes replaced in step 3 plus one.

Q.E.D.

As an example, if the tour is given as in Figure (2.5), then the sequence of nodes obtained in step 1 is

$$1, p_1, 2, p_3, 3, 4, 5, 6, p_2, 1$$

In step 2, the first node (node 1) and p_2 will be discarded from the sequence to give

$$p_1, 2, p_3, 3, 4, 5, 6, 1$$

In step 3, p_1 and p_3 will be replaced by 1, and the sequence would look like

$$1, 2, 1, 3, 4, 5, 6, 1$$

The corresponding 2-tour is that of Figure (2.4).

CHAPTER 3

THE INTEGER PROGRAMMING FORMULATION AND ITS RELAXED LINEAR PROGRAM

3.1 0-1 Programming Formulation

In 1954, Dantzig, Fulkerson and Johnson[3] formulated the traveling salesman problem as:

(IP1):

$$\min \sum_{(i,j)\in A} d_{ij} x_{ij}$$

$$\text{s.t. } (1) \sum_{\substack{j\in N \\ j\neq i}} x_{ij} = 1 \qquad \text{for every node } i \text{ in } N$$

$$(2) \sum_{\substack{i\in N \\ i\neq j}} x_{ij} = 1 \qquad \text{for every node } j \text{ in } N$$

$$(3) \sum_{i\in X} \sum_{j\in \bar{X}} x_{ij} \geq 1 \qquad \text{for every subset } X \text{ of } N \text{ with } 2 \leq |X| \leq |N| - 2$$

$$(4) \; x_{ij} \in \{0,1\} \qquad \text{for every arc } (i,j) \text{ in } A$$

where $\bar{X}$ is the complement of X in N, $|X|$ is the number of elements in X, and where the 0-1 variable x_{ij} is interpreted as:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

The integrality condition (4) requires that each arc of the graph must be either in the tour or not in the tour, the equality constraints (1) and (2) require each node to be passed through exactly once along the tour, and the inequality constraint (3) excludes subtours from the feasible set. Constraint (3) is called the loop constraint.

There are only n equations in (1) and (2), respectively, (where n is the number of nodes in the graph), but the number of inequalities in (3) is given by:

$$\binom{n}{2} + \binom{n}{3} + \ldots + \binom{n}{n-2} = 2^{n-1} - 2n - 2$$

which is too large to be handled even as a linear program for a moderate value of n. This has been the major obstacle to any attempt to solve (IP1).

The first attempt to solve (IP1) was made by Dantzig, et al. for a 42-node problem.

> They overcame the large number of loop constraints by beginning with only a few, and then adding new ones only as they were needed to block subtours. Combinatorial arguments were used to eliminate fractional solutions and to find an optimal tour. Finally, it was demonstrated that for the problem at hand, an ordinary linear program could be devised whose solution gave integer valued x_{ij}'s representing the optimal tour. (Bellmore and Nemhauser[2])

In 1966, Martin[12] reported having solved (IP1) for Dantzig's 42-node problem by a cutting plane algorithm.

> In solving the 42-node problem, Martin began with 84 constraints, 42 of them coming from the assignment problem*, and 42 others being judiciously selected loop constraints in which X contained two nodes. To eliminate fractional solutions, Martin used his "Accelerated Euclidean Algorithm." This algorithm employs Gomory's cutting planes, but in a somewhat different manner than in the Gomory's algorithm. (Bellmore and Nemhauser[2])

Eastman[4] in 1958 and Shapiro[14] in 1966 took an indirect approach by solving the associated assignment problem with a branch-and-bound scheme in order to take care of loop constraints.

---

* If the loop constraints are dropped from (IP1), then the resulting integer program is called the assignment problem, for which there is an efficient algorithm.

> The initial bound is found by solving the associated assignment problem. The bound is taken as the value of the solution to the assignment problem. If the solution to the assignment problem is not a feasible solution to the traveling salesman problem, because of the subtour, then one branches into k subproblems*, where k is the number of arcs in one of the subtours.... The k new assignment problems are solved and determine the lower bounds for all the tours in their respective subsets. If any of these k solutions are tours and if the distance of one of these tours is less than or equal to the lower bounds on the other subsets, then that tour is optimal. If not, then one takes the subset with the lowest bound and branches again according to the subtours present in that solution. Eventually, one is assured of finding an optimal tour. (Bellmore and Nemhauser[2])

There is a different form of loop constraints made by Miller, Tucker, and Zemlin[13] which is:

$$u_i - u_j + n\,x_{ij} \leq n-1 \qquad \text{for all } i \text{ and } j \text{ where } i \neq j,\ j \neq 1$$

$$u_i \geq 0 \text{ and integer} \qquad \text{for all } i$$

The above form of the loop constraint contains fewer inequalities than that of (IP1), but they are not any stronger nor manageable in size.

It is the purpose of this dissertation to develop an algorithm to solve (IP1). Here, we will consider a symmetric problem only, because the computational aspects of an asymmetric problem are similar to those of a symmetric problem.

In the symmetric traveling salesman problem, the reverse of a tour is also a tour with the same distance because $d_{ij} = d_{ji}$ for every pair of nodes i and j. Therefore, the symmetric traveling salesman problem is equivalent to finding a minimal distance tour on an undirected graph

---

* In each subproblem the distance of one of the k arcs is set to infinity in order to prohibit that arc from entering the tour.

G(N,E;D), and was also formulated by Dantzig, et al. as:

(IP2):

$$\min \quad \Sigma_{(i,j)\in E} d_{ij} x_{ij}$$

$$\text{s.t.} \quad \Sigma_{(i,j)\in(X,\bar{X})} x_{ij} = 2 \qquad \text{for every subset X of N with } |X| = 1 \text{ or } |X| = n-1$$

$$\Sigma_{(i,j)\in(X,\bar{X})} x_{ij} \geq 2 \qquad \text{for every subset X of N with } 1 < |X| < n-1$$

$$x_{ij}\in\{0,1\} \qquad \text{for every edge (i,j) in E}$$

where $(X,\bar{X})$ is a set of edges with one end in X and the other end not in X. This set of edges is called a cutset.

The total number, S, of constraints in (IP2) is given by:

$$S = \frac{\binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n-1}}{2} = 2^{n-1} - 1$$

3.2 The Relaxed Linear Program

Suppose there are n nodes and m edges in an undirected graph G(N,E;D). Define an edge-cutset incidence matrix (a_{ij}) by:

$$a_{ij} = \begin{cases} 1 \text{ if edge i is in the cutset j} \\ 0 \text{ otherwise} \end{cases}$$

Here, we assume that the cutsets are numbered such that cutset 1 is the set of edges incident with node 1, cutset 2 is the set of edges incident with node 2, ..., cutset n is the set of edges incident with node n. Then the first n columns of (a_{ij}) are identical to an edge-node incidence matrix formed by:

$$a_{ij} = \begin{cases} 1 & \text{if edge } i \text{ is incident with node } j \\ 0 & \text{otherwise} \end{cases} \qquad (3\text{-}1)$$

for $i = 1, 2, \ldots, m$, and $j = 1, 2, \ldots, n$.

Problem (IP2) of the previous section can be rewritten in terms of the edge-cutset incidence matrix as:

(IP2'):

$$\min \quad \Sigma_{i=1}^{m} x_i d_i$$

$$\text{s.t.} \quad (1) \quad \Sigma_{i=1}^{m} x_i a_{ij} = 2 \qquad \text{for } j=1, 2, \ldots, n$$

$$(2) \quad \Sigma_{i=1}^{m} x_i a_{ij} \geq 2 \qquad \text{for } j = n+1, n+2, \ldots, S$$

$$(3) \quad x_i \in \{0,1\} \qquad \text{for } i = 1, 2, \ldots, m$$

where d_i is the distance of edge i, and the 0-1 variable x_i indicates whether edge i is in the tour.

To solve (IP2') we will first solve its relaxed linear program. The relaxed linear program is obtained by replacing the integrality requirement (3) by $0 \leq x_i \leq 1$ for $i = 1, 2, \ldots, m$. But the following theorem says that the upper bounding constraints need not be written explicitly.

Theorem 3.1

If $x_i > 1$ for some i and if constraint (1) of (IP2') is satisfied, then at least one of the constraints in (2) is not satisfied.

(Proof)

Let us suppose that $x_1 > 1$, and that node 1 and node 2 are the end nodes of edge 1. Further, let X be a set consisting of nodes 1 and 2. Then

$$\Sigma_{i\varepsilon(X,\bar{X})}\ x_i = \Sigma_{i=1}^{m} x_i\, a_{i1} + \Sigma_{i=1}^{m} x_i\, a_{i2} - 2\, x_1$$

$$= 2 + 2 - 2\, x_1 < 2$$

Q.E.D.

Therefore the relaxed linear program of (IP2') can be written as:

(LP):

$$\min \quad \Sigma_{i=1}^{m} d_i\, x_i$$

$$\text{s.t.} \quad (1) \quad \Sigma_{i=1}^{m} x_i\, a_{ij} = 2 \qquad \text{for } j = 1, 2, \ldots, n$$

$$(2) \quad \Sigma_{i=1}^{m} x_i\, a_{ij} \geq 2 \qquad \text{for } j = n+1, n+2, \ldots, S$$

$$(3) \quad x_i \geq 0 \qquad \text{for } i = 1, 2, \ldots, m$$

3.3 The Dual Linear Program*

The dual of (LP) is given by:

(DP):

$$\max \quad Z = 2\, \Sigma_{j=1}^{S} w_j$$

$$\text{s.t.} \quad (1) \quad \Sigma_{j=1}^{S} a_{ij} w_j + w_{S+i} = d_i \qquad \text{for } i = 1, 2, \ldots, m$$

$$(2) \quad w_j \gtreqless 0 \qquad \text{for } j = 1, 2, \ldots, n$$

$$(3) \quad w_j \geq 0 \qquad \text{for } j = n+1, n+2, \ldots, S$$

$$(4) \quad w_{S+i} \geq 0 \qquad \text{for } i = 1, 2, \ldots, m$$

where $w_{S+1}, w_{S+2}, \ldots, w_{S+m}$ are slack variables.

---

* The proofs of the statements made in this section can be found in any linear programming text book, such as Hadley[8].

Let a cost vector $c = (c_1, c_2, \ldots, c_S, c_{S+1}, \ldots, c_{S+m})$ be defined by:

$$c_j = \begin{cases} 2 & \text{for } j = 1, 2, \ldots, S \\ 0 & \text{for } j = S+1, S+2, \ldots, S+m \end{cases} \qquad (3\text{-}2)$$

and write the dual objective function as:

$$Z = \Sigma_{j=1}^{S+m} c_j w_j$$

Let B be the basis matrix at each primal simplex iteration of (DP), and let R be the collection of nonbasic columns of (a_{ij}). B and R will also be used as index sets and as superscripts to denote the basic and nonbasic variables, costs, or columns of (a_{ij}). Then the objective function of (DP) can be written as:

$$Z = c^B w^B - \Sigma_{j \in R} (c^B B^{-1} \bar{a}_j - c_j) w_j$$

and constraint (1) of (DP) can be written as:

$$w^B = B^{-1} d - \Sigma_{j \in R} B^{-1} \bar{a}_j w_j$$

where w^B denotes a vector of basic variables with a corresponding cost vector c^B, and where $\bar{a}_j$ is the j^{th} column of (a_{ij}).

Let us define an m-column vector $\bar{y}_j$ and a real number $z_j - c_j$ associated with each variable w_j by:

$$\bar{y}_j = B^{-1} \bar{a}_j$$

$$z_j - c_j = c^B B^{-1} \bar{a}_j - c_j$$

and rewrite (DP) in terms of $\bar{y}_j$ and $z_j - c_j$ as:

$$\max \quad Z = c^B w^B - \Sigma_{j\in R} (z_j - c_j) w_j$$

$$\text{s.t. } w^B = B^{-1} d - \Sigma_{j\in R} \bar{y}_j w_j$$

$$w_j \gtreqless 0 \qquad \text{for } j = 1, 2, \ldots, n$$

$$w_j \geqq 0 \qquad \text{for } j = n+1, n+2, \ldots, S, S+1, \ldots, S+m$$

At each iteration with basis B the basic and nonbasic variables are given by:

$$w^B = B^{-1} d, \qquad w_j = 0 \text{ for } j\in R$$

and there are the following three cases in which one can improve the basis with a possible increase in the dual objective value:

(i) if $z_j - c_j \neq 0$ for some $j\in R$, such that w_j is an unrestricted variable, then introduce w_j into the basis,

(ii) if $z_j - c_j < 0$ for some $j\in R$, such that w_j is a nonnegative and nonslack variable, then introduce w_j into the basis,

(iii) if $z_j - c_j < 0$ for some $j\in R$, such that w_j is a slack variable, then introduce w_j into the basis.

On the other hand, if $z_j - c_j = 0$ for all $j\in R$, such that w_j is an unrestricted variable, and if $z_j - c_j \geqq 0$ for all $j\in R$, such that w_j is a nonnegative variable, then the present basis is optimal, in which case the optimal solution of (LP) is given by:

$$x = c^B B^{-1} \tag{3-3}$$

If w_j is a basic variable, say the i^{th} basic variable, then the i^{th} column of B is equal to $\bar{a}_j$. Therefore,

$$B^{-1} \bar{a}_j = \bar{e}_i$$

$$c_i^B = c_j$$

where $\bar{e}_i$ is a unit column vector in which the i^{th} element is equal to 1, and where c_i^B denotes the cost associated with the i^{th} basic variable. Hence, it follows that

$$z_j - c_j = c^B B^{-1} \bar{a}_j - c_j = c_i^B - c_j = 0 \qquad \text{for every } j \epsilon B$$

Since we are interested in finding a non-zero $z_j - c_j$ in case (i) and a negative $z_j - c_j$ in cases (ii) and (iii), it is unnecessary to distinguish whether $j \epsilon R$ or $j \epsilon B$.

To check whether there is a positive $z_j - c_j$ in case (i), we compute:

$$z_k - c_k = \max \{z_j - c_j;\ j = 1, 2, \ldots, n\} \tag{3-4}$$

If $z_k - c_k$ is positive, then w_k replaces the r^{th} basic variable, where r is determined from:

$$\frac{w_r^B}{y_{rk}} = \min\left\{\frac{w_i^B}{y_{ik}};\ y_{ik} > 0,\ w_i^B \text{ is a nonnegative variable}\right\} \tag{3-5}$$

where y_{ik} is the i^{th} element of $\bar{y}_k$ and w_i^B is the i^{th} basic variable.

To check whether there is a negative $z_j - c_j$ for cases (i) and (ii), we compute*:

$$z_k - c_k = \min \{z_j - c_j;\ j = 1, 2, \ldots, S\} \tag{3-6}$$

If $z_k - c_k$ is negative, then w_k replaces the r^{th} basic variable, where r is determined from:

$$\frac{w_r^B}{y_{rk}} = \max\left\{\frac{w_i^B}{y_{ik}};\ y_{ik} < 0,\ w_i^B \text{ is a nonnegative variable}\right\} \tag{3-7}$$

---

* An efficient method of computation will be discussed in the next section.

Finally, to check whether there is a negative $z_j - c_j$ for case (iii), we compute:

$$z_{S+k} - c_{S+k} = \min \{z_{S+i} - c_{S+i} \; ; \; i = 1, 2, \ldots, m\} \qquad (3\text{-}8)$$

If $z_{S+k} - c_{S+k}$ is negative, then the k^{th} slack variable replaces the r^{th} basic variable where r is determined from*:

$$\frac{w_r^B}{B_{rk}^{-1}} = \max \left\{ \frac{w_i^B}{B_{ik}^{-1}} \; ; \; B_{ik}^{-1} < 0, \; w_i^B \text{ is nonnegative variable} \right\} \qquad (3\text{-}9)$$

where B_{ik}^{-1} is the $(ik)^{th}$ element of the inverse of the basis matrix.

When the r^{th} basic variable is replaced by w_k, the pivot operation to obtain the inverse of the improved basis matrix is given by:

$$\begin{cases} \hat{B}_{ij}^{-1} = B_{ij}^{-1} - \dfrac{B_{rj}^{-1}}{y_{rk}} \, y_{ik} & \text{for } i = 1, 2, \ldots, m, \; i \neq r, \\ & \text{and } j = 1, 2, \ldots, m \\ \hat{B}_{rj}^{-1} = \dfrac{B_{rj}^{-1}}{y_{rk}} & \text{for } j = 1, 2, \ldots, m \end{cases}$$

The corresponding new basic variables are given by:

$$\begin{cases} \hat{w_i^B} = w_i^B - \dfrac{w_r^B}{y_{rk}} \, y_{ik} & \text{for } i = 1, 2, \ldots, m, \; i \neq r \\ \hat{w_r^B} = \dfrac{w_r^B}{y_{rk}} \end{cases}$$

where if w_k is the k^{th} slack variable, then $\bar{y}_k$ is equal to the k^{th} column of B^{-1}.

---

* The column $\bar{a}_{S+k}$ associated with the k^{th} slack variable is equal to $\bar{e}_k$. Therefore, $\bar{y}_k$ is equal to the k^{th} column of B^{-1}.

If r cannot be determined from (3-5) by having $y_{ik} \leq 0$ for all i such that w_i^B is a nonnegative variable, then (DP) has an unbounded solution, which implies that (LP) has no feasible solution. The same argument follows for (3-7) and (3-9).

3.4 A Column Generation Scheme

In this section we discuss how the maximization or the minimization problems (3-4), (3-6), and (3-8) can be solved.

We know that c_j's are given by (3-2), and that

$$\bar{a}_{S+i} = \bar{e}_i \qquad \text{for } i = 1, 2, \ldots, m$$

Let an m-row vector σ be defined by:

$$\sigma = c^B B^{-1} \tag{3-10}$$

Then $z_j - c_j$ can be written in terms of σ as:

$$z_j - c_j = \Sigma_{i=1}^{m} \sigma_i a_{ij} - 2 \qquad \text{for } j = 1, 2, \ldots, S \tag{3-11}$$

$$z_{S+i} - c_{S+i} = \sigma_i \qquad \text{for } i = 1, 2, \ldots, m \tag{3-12}$$

Now, equation (3-8) can easily be computed by taking the minimum of σ_i by equation (3-12).

Let σ_i be the capacity of edge i. The edge-cutset incidence matrix (a_{ij}) was defined to coincide with the edge-node incidence matrix for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, where m is the number of edges and n is the number of nodes. Hence, it follows that $\sum_i \sigma_i a_{ij}$ is the total capacity of edges incident with node j. Therefore, (3-4) can be computed by enumerating the total capacity incident with each node, and subtracting 2 from it by equation (3-11).

(3-6) can not be computed by an enumeration because $S = 2^{n-1} - 1$ is too large a number. However, the minimum of $(\Sigma_i \sigma_i a_{ij})$ is "the minimal cut capacity over the graph," which can be obtained by a max-flow algorithm of Ford and Fulkerson[6].

Ford and Fulkerson developed an efficient algorithm to obtain a maximal flow from one node, called the source, to another node, called the sink, subject to capacity restrictions on each arc (or on each edge in the case of an undirected graph). This max-flow algorithm also provides a minimal cut capacity separating the source from the sink*. The minimal cut capacity over the graph is the globally minimal cut capacity which is the minimum of all cut capacities separating a pair of nodes.

Since any cut must separate the home node from some of the customer nodes, the globally minimal cut does also; and this globally minimal cut capacity must be obtained as the minimum of the n-1 max-flow problems, each of which has the home node as a source and one of the n-1 customer nodes as a sink.

The max-flow algorithm of Ford and Fulkerson requires that the capacity on each arc must be nonnegative. In our case, the capacity σ_i is not necessarily nonnegative at each iteration. But if σ_i is negative for some i, then $z_{S+i} - c_{S+i}$ is negative from equation (3-12), and the i^{th} slack variable will be introduced into the basis. Therefore, if case (iii) is checked first, then the capacities will be all nonnegative when the max-flow algorithm is employed.

Finally, σ_i need not be computed from the defining equation (3-10) at

---

* A cut capacity separating the source from the sink is the total capacity of edges in the cutset $(X,\bar{X})$ with the source in X and the sink not in X.

each iteration. There also exists a pivot operation by which σ at the next iteration, denoted by $\hat{\sigma}$, can be obtained from σ at the present interation:

$$\hat{\sigma}_i = \sigma_i - \frac{B^{-1}_{ri}}{y_{rk}} (z_k - c_k) \qquad \text{for } i = 1, 2, \ldots, m$$

where w_k is assumed to be replacing the r^{th} basic variable.

CHAPTER 4

A SEARCH FOR AN ALL-INTEGER SOLUTION

4.1 A Matching Constraint for an Odd Cycle

We could solve (LP) of chapter 3 by a column generation scheme applied to its dual, but (LP) does not necessarily terminate in an all-integer solution.

Theorem 4.1

If an optimal solution x^o of (LP) is not all-integer, then there is a cycle such that:

$$0 < x_i^o < 1 \qquad (4\text{-}1)$$

for every edge i in the cycle.

(Proof)

Suppose $0 < x_k^o < 1$. Let the one end node of edge k be node p, and assume that edge 1, edge 2, and edge 3 are also incident with node p, as is shown in Figure (4.1). Then at least one of x_1^o, x_2^o, x_3^o, say x_1^o, must be $0 < x_1^o < 1$ to satisfy the constraint (1) of (LP), namely,

$$\Sigma_{i=1}^{m} x_i a_{ip} = x_1 + x_2 + x_3 + x_k = 2$$

By continuing this argument with edge 1 and another end node of edge 1, say node q, and so forth, one ends up with a cycle satisfying equation (4-1)

Q.E.D.

A cycle the edges of which satisfy equation (4-1) is called a noninteger cycle.

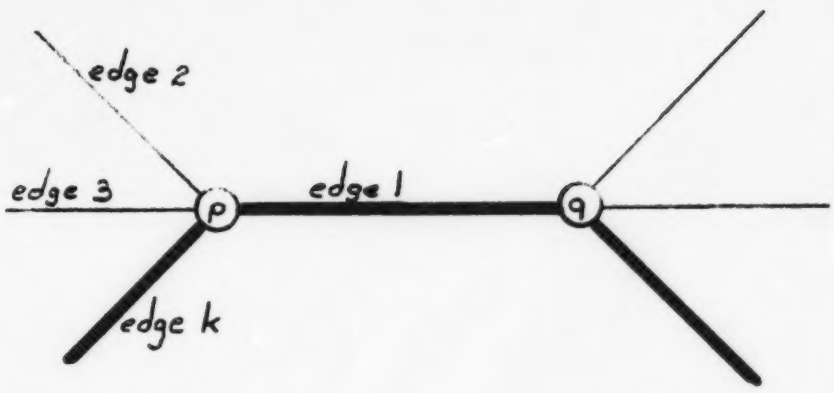


Fig. (4.1) A part of a noninteger solution.

Theorem 4.2

If an optimal solution x^0 of (LP) has a noninteger cycle which contains an odd number of nodes and is disjoint with other noninteger cycles, then it must be the case that $x_i^0 = \frac{1}{2}$ for every edge i in the disjoint noninteger odd cycle.

(Proof)

Suppose edges 1, 2, 3, ..., 2p+1 are in the noninteger odd cycle as is shown in Figure (4.2). Because this cycle is disjoint with other non-integer cycles, we must have:

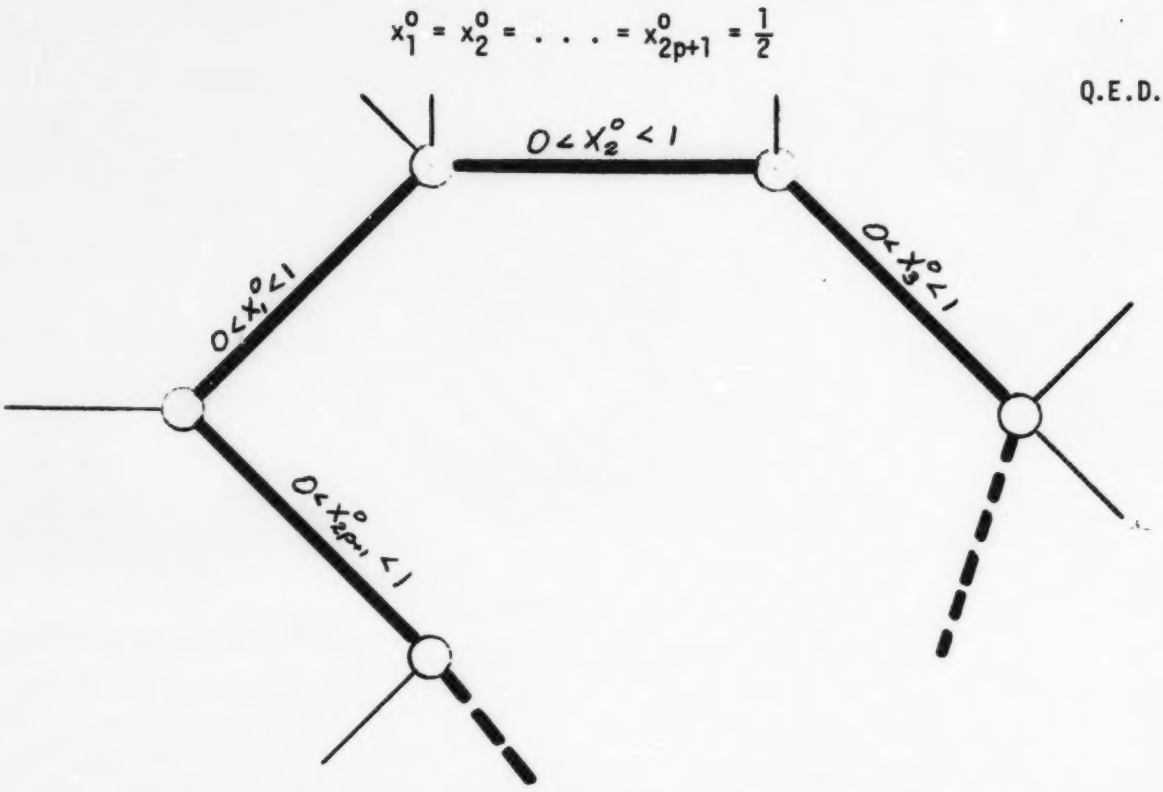
$$x_1^0 + x_2^0 = 1$$

$$x_2^0 + x_3^0 = 1$$

$$\vdots$$

$$x_{2p+1}^0 + x_1^0 = 1$$

from which we get:



Q.E.D.

Fig. (4.2) A noninteger odd cycle of 2p+1 nodes.

Theorem 4.3

If an optimal solution x^o of (LP) contains a disjoint noninteger odd cycle, then

$$\Sigma_{i \in M} x_i^o = r + \frac{1}{2} \qquad (4\text{-}2)$$

where M is a set of edges with both ends in the cycle, plus all t edges with one end node in the cycle and having $x_i^o = 1$, and where r is an integer given by:

$$2r + 1 = 2x \text{ (no of nodes in the cycle)} + t$$

(Proof)

Suppose a noninteger disjoint odd cycle contains 2p+1 nodes. Then there are 2p+1 edges in the cycle and all of them have $x_i^0 = \frac{1}{2}$ by theorem 4.2. Suppose that q edges with both end nodes in the cycle have $x_i^0 = 1$. Then only t edges with one end node in the cycle have $x_i^0 = \frac{1}{2}$, where $t = 2p+1 - 2q$. As an example, if the noninteger disjoint odd cycle is that of Figure (4.3), then $2p+1 = 5$, $q = 1$, $t = 3$. Therefore, we must have:

$$\Sigma_{i \in M} x_i^0 = \frac{1}{2}(2p+1) + q + t$$

$$= (2p+1) - \frac{1}{2}(2p+1 - 2q) + t$$

$$= (2p+1) + \frac{1}{2} t$$

$$= r + \frac{1}{2}$$

Q.E.D.

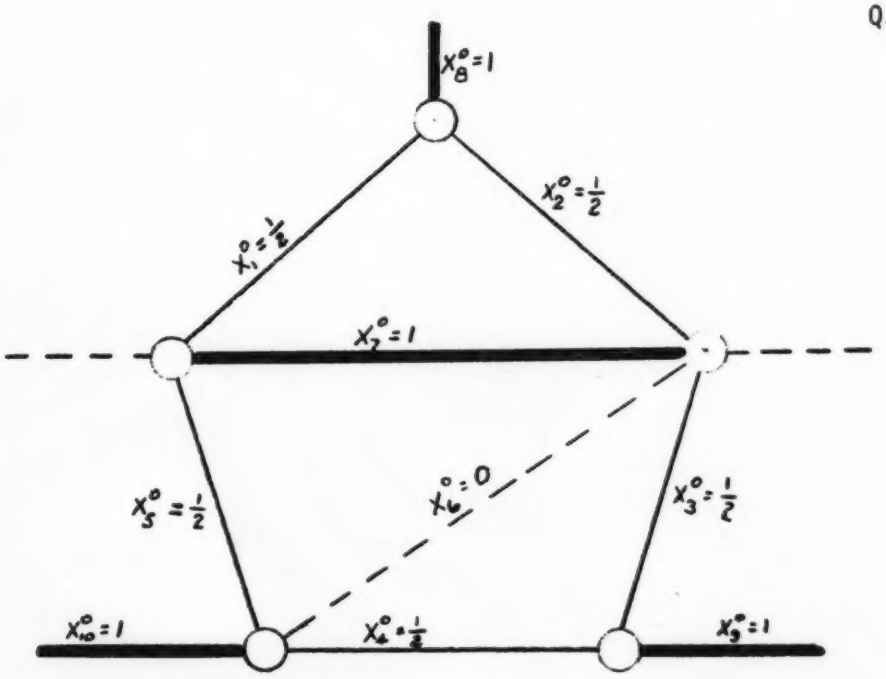


Fig. (4.3) A noninteger odd cycle of 5 nodes.

According to theorem 4.3, if an optimal solution of (LP) is not all-integer, and if there is a noninteger disjoint odd cycle, then by adding an additional constraint:

$$\Sigma_{i\epsilon M} x_i \leq r \qquad (4\text{-}4)$$

to (LP), the noninteger solution can be made infeasible, where r must be determined from equation (4-3). As an example, if the optimal solution x^o of (LP) is given as in Figure (4.3), then inequality (4-4) becomes:

$$x_1 + x_2 + x_3 + \ldots + x_{10} \leq 6$$

whereas:

$$x_1^o + x_2^o + x_3^o + \ldots + x_{10}^o = 6.5$$

In order to use (4-4) as an additional constraint for the traveling salesman problem, one must prove that every tour satisfies (4-4). We will prove it by considering a b-matching problem.

In a b-matching problem, one must pick a set of edges such that not more than b_j edges are incident with each node j and such that the total edge weight (distance) is maximized. For this problem, Edmonds [5] has proved the following theorem:

Generalized Theorem (P)

All the extreme points of polyhedron II consist of integer components.

II: (1) $0 \leq x \leq 1$

(2) $\Sigma_{i\epsilon M} x_i \leq b_j$ for every node j, where V_j is the set of edges incident with node j,

(3) $\Sigma_{i\in M} x_i \leq r$ for all M and r, where r is a positive integer, and where M corresponds to all edges with both ends in a set of nodes, X, plus a number t of edges with one end in X where less than b_j of the t edges meet a node j in X, and where $2r+1 = t + \Sigma_{j\in X} b_j$.

In a tour, there are exactly two edges incident with a node, and every tour is a feasible solution of the b-matching problem with $b_j = 2$ for every node j in the graph. (But not every feasible solution of the 2-matching* problem is a tour.) Therefore, every tour satisfies the constraints II. (1), (2), (3) of the Generalized Theorem (P). Note that inequality (4-4) is a special case of the constraint II. (3) with X as a set of nodes in the disjoint odd cycle, and with t as the number of edges with one end in X having $x_i^o = 1$. Hence, every tour satisfies (4-4), and (4-4) can be used as an additional constraint for the traveling salesman problem.

Of course, one can add all the inequalities of II.(3) to (IP2) in order to make every possible noninteger disjoint odd cycle infeasible. But since a column generation scheme by which all the constraints of II. (3) can be taken care of in the dual has not been found, we will add the necessary ones only after an optimal solution of (LP) is obtained.

An additional constraint in (LP) results in an additional variable in (DP). If the constraint (4-4) is added to (LP), then the column $\bar{a}_k$ associated with the new variable in (DP) is given by:

---

* A b-matching problem with $b_j = 2$ for every node j in the graph is called a 2-matching problem.

$$a_{ik} = \begin{cases} -1 \text{ if edge } i \text{ is in } M \\ 0 \text{ otherwise} \end{cases}$$

where (4-4) is multiplied by (-1) in order to convert it into a "greater than" type of constraint, which is then consistent with other constraints in (LP). The corresponding pricing element $z_k - c_k$ is given by:

$$z_k - c_k = c^B B^{-1} \bar{a}_k - c_k = -\Sigma_{i \in M} x_i - (-r) = -\frac{1}{2} < 0$$

Therefore, at the next iteration, this new column will be the pivot column.

If a noninteger cycle is either node-joint with another noninteger cycle, or if it contains an even number of nodes, then no constraint has been found which makes it infeasible.

4.2. A Branch-And-Bound Scheme

A branch-and-bound scheme is an implicit enumeration method, in which some of the 0-1 variables are fixed at either 0 or 1 at each stage of the enumeration (called the vertex of an enumeration tree), and an improved feasible integer solution is sought using bounding information in order to avoid total enumeration. There are four major parts of this enumeration scheme, namely; Branching, Solving a linear program $(LP)_j$, Fathoming, and Backtracking.

Branching is an assignment of either 0 or 1 to some 0-1 variable; the linear program $(LP)_j$ is a subproblem associated with each enumeration vertex; Fathoming is a decision to drop a certain vertex of enumeration tree from further Branching; and Backtracking is a rule by which a live (unfathomed) vertex can be found. Each will be described in more detail for the traveling salesman problem.

The objective is to solve (IP2), which we will rename as $(IP)_o$ and rewrite in compact form as:

$(IP)_o$:

$$z(IP)_o = \min z(x)$$

$$\text{s.t. } x \in S_o \text{ and integer}$$

where the objective function $z(x)$ is given by

$$z(x) = \Sigma_{i=1}^{m} x_i \, d_i \tag{4-5}$$

and S_o is the constraint set of (LP) of chapter 3.

Associated with each vertex V_j of the enumeration tree are an index set P_j in which a branching history is recorded such as:

$$+i \in P_j \Longrightarrow x_i \text{ is fixed at 1}$$

$$-i \in P_j \Longrightarrow x_i \text{ is fixed at 0}$$

and an integer program $(IP)_j$ given by:

$(IP)_j$:

$$z(IP)_j = \min z(x)$$

$$\text{s.t. } x \in S_j \text{ and integer}$$

where the objective function is the same for all V_j and is given by (4-5), and where S_j is given by:

$$S_j = S_o \cap \left\{ x; \begin{array}{lll} x_i = 1 & \forall\, i & \ni +i \in P_j \\ x_i = 0 & \forall\, i & \ni -i \in P_j \end{array} \right\} \tag{4-6}$$

The relaxed linear program of $(IP)_j$ is denoted by $(LP)_j$ and is given by:

$(LP)_j$:

$$z(LP)_j = \min z(x)$$

$$\text{s.t. } x \in S_j$$

Here, $(LP)_0$ is identical to (LP) of chapter 3.

Branching Rule

Suppose an optimal solution x^j of $(LP)_j$ is not all-integer, and we chose to branch on edge k for some k such that $0 < x_k^j < 1$. Here, x_j^k denotes the k^{th} component of x^j. $(LP)_{j+1}$ is obtained from $(LP)_j$ by adding an additional constraint, $x_k = 0$ or $x_k = 1$. Then the dual of $(LP)_{j+1}$ is obtained by adding an additional variable w_k to the dual of $(LP)_j$.

Column $\bar{a}_k$ associated with w_k in the dual of $(LP)_{j+1}$ is given by $\bar{a}_k = \bar{e}_k$, from which we get:

$$\bar{y}_k = B^{-1} \bar{a}_k = (\text{the } k^{th} \text{ column of } B^{-1})$$

where B^{-1} is the inverse of the optimal basis of $(LP)_j$.

The cost associated with the new variable, w_k, is given by:

$$c_k = \begin{cases} 0 & \text{if branching is } x_k = 0 \\ 1 & \text{if branching is } x_k = 1 \end{cases}$$

and the pricing element is given by:

$$z_k - c_k = c_B B^{-1} \bar{a}_k - c_k = x^j \bar{e}_k - c_k = \begin{cases} x_k^j & \text{if branching is } x_k = 0 \\ x_k^j - 1 & \text{if branching is } x_k = 1 \end{cases}$$

At the next pivot iteration, w_k is introduced into the basis because*

* Note that the constraint arising from the Branching is an equality constraint, and w_k is an unrestricted variable.

$z_k - c_k = 0$, and the change in the dual objective value, Δ_{j+1}, is given by

$$\Delta_{j+1} = \begin{cases} -\dfrac{w_r^B}{B_{rk}^{-1}}\, x_k^j & \text{if branching is } x_k = 0 \\ -\dfrac{w_r^B}{B_{rk}^{-1}}\, (x_k^j - 1) & \text{if branching is } x_k = 1 \end{cases}$$

If Branching is $x_k = 0$, then r must be determined from:

$$\frac{w_r^B}{B_{rk}^{-1}} = \max\left\{ \frac{w_i^B}{B_{ik}^{-1}} \; ; \; B_{ik}^{-1} < 0,\ w_i^B \text{ is a nonnegative variable} \right\} \quad (4\text{-}7)$$

and if Branching is $x_k = 1$, then r must be determined from:

$$\frac{w_r^B}{B_{rk}^{-1}} = \min\left\{ \frac{w_j^B}{B_{ik}^{-1}} \; ; \; B_{ik}^{-1} > 0,\ w_i^B \text{ is a nonnegative variable} \right\} \quad (4\text{-}8)$$

Now, we will branch in order to get a maximum possible change in the dual objective value at the next iteration, that is, to maximize Δ_{j+1}. For that purpose, we first compute:

$$\theta_o(k_o) = \max_k \left\{ -\frac{w_r^B}{B_{rk}^{-1}}\, x_k^j \; ; \; 0 < x_k^j < 1 \right\}$$

where r must be determined from (4-7) for each k. Here, k_o denotes the index on which the maximum is achieved. Then compute:

$$\theta_1(k_1) = \max_k \left\{ - \frac{w_r^B}{B_{rk}^{-1}} \; (x_k^j - 1);\; 0 < x_k^j < 1 \right\}$$

where r must be determined from (4-8) for each k. Here k_1 denotes the index on which the maximum is achieved.

The Branching Rule is given by:

$$x_{k_0} = 0 \quad \text{if } \theta_0(k_0) > \theta_1(k_1)$$

$$x_{k_1} = 1 \quad \text{if } \theta_0(k_0) \leq \theta_1(k_1)$$

Here, the tie is broken arbitrarily.

Finally, if r cannot be determined from (4-7) for some k such that $0 < x_k < 1$ by having $B_{ik}^{-1} \geq 0$ for all i such that w_i^B is a nonnegative variable, then we branch as $x_k = 0$, and fathom V_{j+1} immediately. This is because $(LP)_{j+1}$ has no feasible solution. Also, if r cannot be determined from (4-8) for the same reason, then we branch as $x_k = 1$, and fathom V_{j+1} immediately.

Solving the Linear Program $(LP)_j$

Theorem 4.4

Let $|P_j^+|$ be the number of positive elements of P_j. Then

$$S_j = S_0 \cap \left\{ x;\; \Sigma_{+i \in P_j} \; x_i = |P_j^+|,\; \Sigma_{-i \in P_j} \; x_i = 0 \right\}$$

(Proof)

If $x \in S_0$, then:

(i) $x_i = 1 \; \forall_i \; \exists_{+i\varepsilon P_j} \quad (\Longleftrightarrow) \quad \Sigma_{+i\varepsilon P_j} \; x_i = |P_j^+|$

(ii) $x_i = 0 \; \forall_i \; \exists_{-i\varepsilon P_j} \quad (\Longleftrightarrow) \quad \Sigma_{-i\varepsilon P_j} \; x_i = 0$

because $0 \leq x_i \leq 1$ for all i.

Q.E.D.

By theorem 4.4, $(LP)_j$ has at most two more constraints than $(LP)_o$, namely:

$$\Sigma_{+i\varepsilon P_j} \; x_i = |P_j^+| \tag{4-9a}$$

$$\Sigma_{-i\varepsilon P_j} \; x_j = 0 \tag{4-9b}$$

Therefore, the dual of $(LP)_j$ can be solved from the optimal tableau of the dual of $(LP)_o$ by adding at most two more columns, corresponding to (4-9a) and (4-9b). It is unnecessary to store all the optimal tableaux of $(LP)_j$'s.

The pricing element in the dual, associated with constraint (4-9a) or (4-9b) in $(LP)_j$, is given by:

$$z_i - c_k = \begin{cases} \Sigma_{+i\varepsilon P_j} \; x_i^o - |P_j^+| & \text{for the dual variable corresponding to (4-9a)} \\ \Sigma_{-i\varepsilon P_j} \; x_i^o & \text{for the dual variable corresponding to (4-9b)} \end{cases}$$

where x^o is the optimal solution of $(LP)_o$.

Fathoming Test

Let $\overline{z}_j$ and $\underline{z}_j$ be an upper and a lower bound of $(IP)_j$, satisfying:

$$\underline{z}_j \leq z(IP)_j \leq \overline{z}_j$$

The lower bound can be obtained from $(LP)_j$ as:

$$\underline{z}_j = z(LP)_j$$

Any feasible integer solution of $(LP)_j$ can be an upper bound, but if an integer feasible solution is not available, then we simply have:

$$\overline{z}_j = \infty$$

There are two cases in which a vertex V_j can be fathomed, namely:

(i) $\underline{z}_j = \overline{z}_j$

(ii) $\underline{z}_j \geq \overline{z}_0$

In case (i), an optimal integer solution of $(IP)_j$ has been found, which is the best of all successors of V_j. In case (ii), any successor of V_j cannot provide a better solution than $\overline{z}_0$, and can be dropped from further enumeration.

In actual computation, $\overline{z}_0$ will be constantly updated so that it represents the best known integer feasible solution of $(IP)_0$. If the dual objective value of $(LP)_j$ is greater than or equal to $\overline{z}_0$ even before the termination, then V_j can be fathomed, because the dual objective value is monotonically non-decreasing as the iteration continues.

If $(IP)_j$ has no feasible solution, then we have $\underline{z}_j = \infty$, and V_j can be fathomed by case (ii). Therefore, before $(LP)_j$ is solved, it is desirable to check the following infeasibilities:

(a) If $\Sigma_{+i\varepsilon P_j} a_{ik} > 2$ for some $k = 1, 2, \ldots, n$, then $(IP)_j$ has no feasible solution.

(b) If $\Sigma_{i=1}^{m} a_{ik} - \Sigma_{-i\varepsilon P_j} a_{ik} < 2$ for some k = 1, 2, . . ., n, then $(IP)_j$ has no feasible solution.

(c) If a set of edges, which are in P_j on a positive level, forms a subtour, then $(IP)_j$ has no feasible solution.

(d) Let E' be a subset of E defined by:

$$E' = E - \{\text{edges in } P_j \text{ on a negative level}\}$$

If there is no tour in a graph G(N,E'), then $(IP)_j$ has no feasible solution. As an example, if G(N,E) is given as in Figure (4.4a), and if $P_j = \{+3, -1, -2,\}$, then G(N,E') is given as in Figure (4.4b).

Conditions (a) and (b) can be checked by enumeration, and (c) can be checked by moving along the edges which are fixed at 1 by Branching. Finally, (d) cannot be checked thoroughly unless a traveling salesman problem is solved on G(N,E'). But some of the special cases can be checked by the following:

Let E_k be a subset of E' defined by:

$$E_k = E' - \{\text{edges incident with node k}\}$$

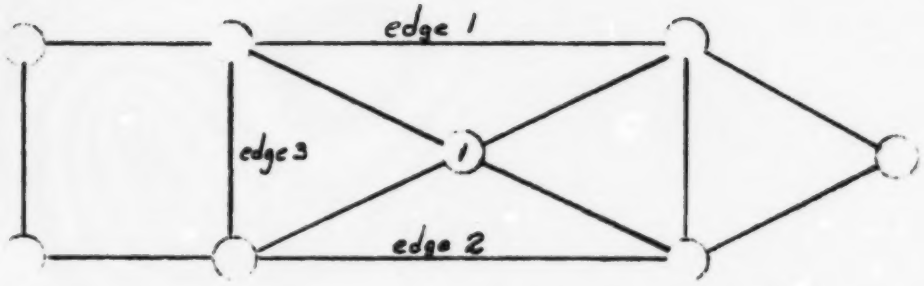
and let N_k be given by:

$$N_k = N - \text{node k} \qquad \text{for k} = 1, 2, \ldots, n$$

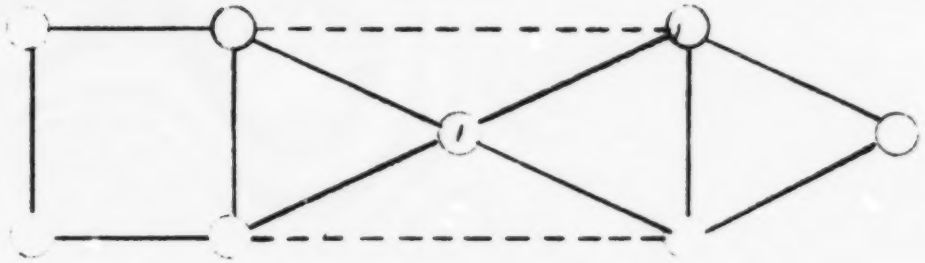
If the graph $G(N_k,E_k)$ is disconnected, then there is no tour in G(N,E'). As an example, if G(N,E') is given as in Figure (4.4b), then $G(N_1,E_1)$ is given as in Figure (4.4c). The disconnectedness of the graph $G(N_k,E_k)$ can easily be checked by labeling the nodes in N_k which are connected to an arbitrarily fixed node by some edges in E_k.

Backtracking

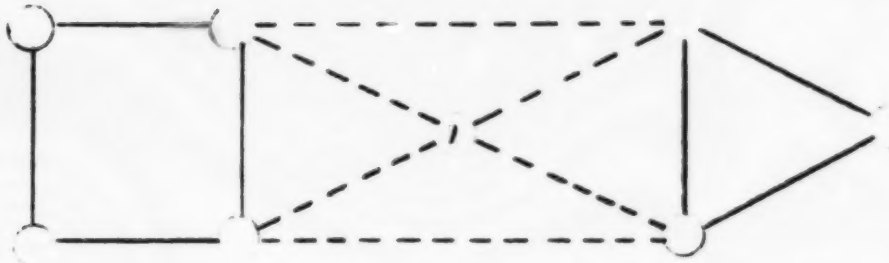
Backtracking is a rule by which a live vertex can be reached from the



(a) A graph G(N,E)



(b) A graph G(N,E') obtained from G(N,E) of (a) when P_j = +3,-1,-2



(c) A graph $G(N_1,E_1)$ obtained from G(N,E') of (b)

Fig. (4.4)

just fathomed vertex. Let us arrange the elements of P_j in order of being branched, and distinguish each element in four different ways according to the branching on each element, namely:

(a) +i; if edge i is branched as $x_i = 1$, and if $x_i = 0$ has not been considered,

(b) +$\underline{i}$; if edge i is branched as $x_i = 1$, and if $x_i = 0$ has been fathomed,

(c) -i; if edge i is branched as $x_i = 0$, and if $x_i = 1$ has not been considered,

(d) -$\underline{i}$; if edge i is branched as $x_i = 0$, and if $x_i = 1$ has been fathomed.

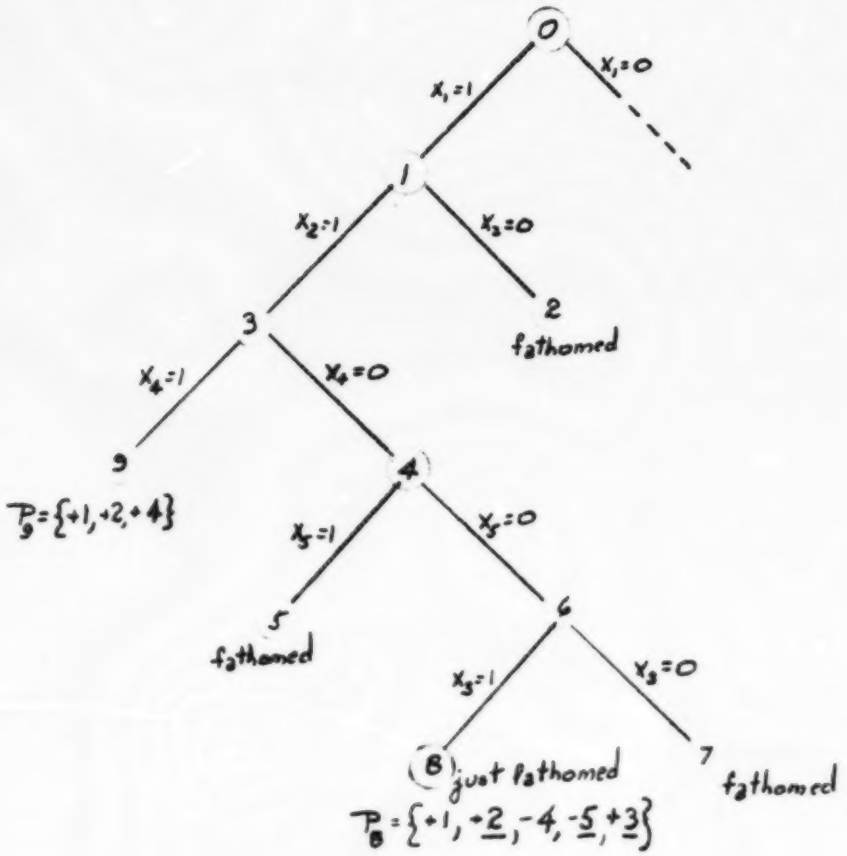


Fig. (4.5) An enumeration tree.

Suppose V_j is just fathomed. P_{j+1}, associated with the next vertex V_{j+1}, is obtained from P_j as follows:

> Replace the last element without an underline in P_j by its opposite sign, and then underline it; then, drop all of its successor elements from P_j.

As an example, let us consider the enumeration tree of Figure (4.5). Suppose V_8 has just been fathomed. P_8 is given by:

$$P_8 = \{+1, +\underline{2}, -4, -\underline{5}, +\underline{3}\}$$

The last element without an underline is -4, and it must be replaced by $+\underline{4}$. Then, the last two elements of P_8 must be discarded, and P_9 would be:

$$P_9 = \{+1, +\underline{2}, +\underline{4}\}$$

The following steps constitute a summary of the branch-and-bound scheme:

(Step 1) Initialization

> An optimal solution of $(LP)_o$ is obtained and it is not all-integer.
>
> Let $\underline{z}_o = z(LP)_o$, $\overline{z}_o = \infty$, $P_o = \emptyset$.
>
> Let $j = 1$ and go to step 2.

(Step 2) Branching

> Get the branching element x_k.
>
> Let $P_j = \begin{cases} \{P_o, +k\} \text{ if branching is } x_k = 1 \\ \{P_o, -k\} \text{ if branching is } x_k = 0 \end{cases}$
>
> Go to step 3.

(Step 3) Fathoming Test 1

Check infeasibilities (a), (b), (c), and (d).

If $(IP)_j$ has no feasible solution, then fathom V_j, and go to step 6.

Otherwise, go to step 4.

(Step 4) Solving $(LP)_j$ and updating the bounds.

Solve $(LP)_j$.

If $(LP)_j$ has no feasible solution then fathom V_j and go to step 6.

If the optimal solution of $(LP)_j$ is all-integer, then let

$$\overline{z}_o = \min (\overline{z}_o, z(LP)_j)$$

Let $\underline{z}_j = z(LP)_j$ and go to step 5.

(Step 5) Fathoming Test 2

If $\underline{z}_j \geq \overline{z}_o$, then fathom V_j, and go to step 6.

Otherwise, let $j = j + 1$, and go to step 1.

(Step 6) Backtracking

V_j is fathomed.

Get P_{j+1} from P_j.

If $P_{j+1} = \emptyset$, then terminate.

Otherwise, go to step 1 with $j = j + 1$.

CHAPTER 5

COMPUTATIONAL RESULTS AND CONCLUSIONS

The algorithm proposed in the previous chapters was coded in Fortran IV on the UNIVAC 1108. The inverse of the basis matrix is stored in core.

The sample problems were randomly generated Euclidean graphs, for which the x- and y-coordinates of each node were picked from a uniform continuous distribution between 0 and 100, and the Euclidean distances between every pair of nodes computed from these coordinates.

The results are shown in Table 5.1 below:

No. of nodes	No. of problems	Avg. run time (seconds)	Avg. no. of pivot oper.	Max. no. of pivot oper.
10	20	1.1	21.5	31
15	20	5.4	37.4	50
20	20	24.4	57.9	89

Table 5.1 Computational Results

The reported time includes the time spent for the random generation of the Euclidean graph. Out of sixty problems with three different sizes, the linear program (LP), with a matching constraint, terminated all-integer for fifty nine problems. For one problem with 20 nodes which did not terminate in all-integer variables, only one branching was needed to obtain an optimal tour.

The algorithm has been tested for symmetric problems only, although an asymmetric problem can be solved as well.

When solving the linear program, the size of the basis matrix is equal to the number of edges in the graph. If the problem is very large compared to the size of the available core memory, a product form of the inverse, accompanied by a preliminary procedure to reduce the number of edges, should be employed.

M. Held (IBM) has suggested in a conference that a tailing off* phenomenon might be associated with the column generation scheme of chapter 4. Adequate computing facilities have not been available to test this hypothesis.

---

*"Tailing off" is the case in which the convergence of the algorithm is so slow that iterations continue near the optimum, but termination does not occur for a very long time.

REFERENCES

1. Bellmore, M., and J. C. Malone, "Pathology of Traveling Salesman Subtour Elimination Algorithm," *Operations Research, 19*, 278-307 (1971).
2. Bellmore, M., and G. L. Nemhauser, "The Traveling Salesman Problem; A Survey," *Operations Research, 16*, 538-558 (1968).
3. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "Solution of Large Scale Traveling Salesman Problem," *Operations Research, 2*, 393-410 (1954).
4. Eastman, W. L., "Linear Programming with Pattern Constraints," Ph.D. Dissertation, Harvard University (1958).
5. Edmonds, J., "Maximum Matching and a Polyhedron with 0-1 Vertices," *Jour. Research, National Bureau of Standards, 69B*, 125 (1965).
6. Ford, R., and D. R. Fulkerson, *Flows in Networks*, Princeton University Press (1962).
7. Gilmore, P. C., and R. E. Gomory, "Sequencing a One-State Variable Machine: A Solvable Case of the Traveling Salesman Problem," *Operations Research, 12*, 655-679 (1964).
8. Hadley, G., *Linear Programming*, Addison Wesley (1962).
9. Hardgrave, W. W., and G. L. Nemhauser, "On the Relation Between the Traveling Salesman Problem and the Longest Path Problem," *Operations Research, 10*, 647-657 (1962).
10. Held, M., and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees," *Operations Research, 18*, 1138-1162 (1970).

11. Held, M., and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," *Math. Prog.*, *1*, 6-25 (1971).

12. Martin, G. T., "Solving the Traveling Salesman Problem by Integer Linear Programming," CEIR, New York (1966).

13. Miller, C. E., A. W. Tucker, and R. A. Zemlin, "Integer Programming Formulations and Traveling Salesman Problem," *Jour. ACM*, *7*, 326-329 (1960).

14. Shapiro, D., "Algorithm for the Solution of the Optimal Cost Traveling Salesman Problem," Sc.D. Thesis, Washington University, St. Louis (1966).